



PSEUDO-RANDOM NUMBER GENERATOR

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to a method of and apparatus for generating pseudo-random numbers.

2. Description of the Prior Art

Pseudo-random numbers are used for a variety of purposes including simulation studies, information processing, communication, and encryption. Pseudo-random number generators create sequences of values that appear to have been generated by random processes even though the sequences are not truly random. The results of a pseudo-random number process should be adequately distributed across the desired range of possible numbers so as to mimic the results that might have come from a truly random process. Pseudo-random results should not exhibit discernable patterns or other observable relationships between the observable output values that would make prediction or other analysis of the observable output sequence possible.

The search for pseudo-random number generators that satisfy the above conditions has yielded a number of interesting and useful processes. The linear feedback shift register (LFSR) process is easy to implement and has been widely used but has an inherent weakness due to the strict linearity of its processes. Another widely used generator is the classical linear congruential generator (LCG), represented as $x_n = (ax_{n-1} + b) \bmod m$, where x is the output series, x_0 is the seed value, and a , b , and m are constants. For example, the LCG process is the framework used by DeVane in the high-speed pseudo-random number generator of U.S. Patent #5,187,676, by Finkelstein in the encryption protection in a communication system of U.S. Patent #6,014,446, by Tiedemann *et al.* in

1 the system for testing a digital communication channel of U.S. Patent #5,802,105, by
2 Ridenour in the high precision pseudo-random number generator of U.S. Patent
3 #5,541,996, and by Shimada in the pseudo-random number generator of U.S. Patent
4 #6,097,815. LCG-based systems can generate well mixed numbers and will pass certain
5 statistical tests, although the sequence generated by an LCG typically can be inferred
6 even if the constant parameters a , b , m and the seed x_0 are all unknown.

7
8 The multiple recursive generator (MRG) is similar to an LCG but extends the range of
9 the recursion from the immediately preceding output value to more distantly produced
10 ones. The MRG process can be represented as $x_n = (a_1 x_{n-1} + \dots + a_k x_{n-k}) \bmod m$, where a_1
11 $\dots a_k$ and m are constants. Lagged Fibonacci generators and some combined generators
12 are essentially MRGs. The LCG process also has been extended to additional
13 dimensions to create a matrix method (MM) process represented as $X_n = (AX_{n-1}) \bmod m$
14 where X is a vector of output values and A is a constant transition matrix. Niederreiter
15 introduced the multiple-recursive matrix method (MRMM) as a framework for
16 encompassing essentially all of the linear methods described above as well as several
17 others such as the Generalized Feedback Shift Register (GFSR) and the “twisted” GFSR.
18 A good example of the twisted GFSR is the recently developed Mersenne Twister
19 described by Matsumoto and Nishimura. The general form of the MRMM process is X_n
20 $= (A_1 X_{n-1} + \dots + A_k X_{n-k}) \bmod m$, where $A_1 \dots A_k$ and m are constants.

21
22 In these conventional systems, the modulus operator is typically chosen to be a fixed
23 number, which may be determined by the hardware constraints of the computer systems
24 to be used. Often, the word length is a critical factor; for instance, 2^{32} is typically chosen
25 as the modulus value for 32-bit computer systems. Using a fixed modulus simplifies the
26 determination of the output range of pseudo-random number generator. A fixed modulus
27 of 2^{32} , for example, creates a range of actual output values from 0 to $2^{32} - 1$. Others, such
28 as Shimada, have suggested varying the modulus operator by using a set of prime
29 numbers (see U.S. Patent #6,097,815). Shimada uses a three-part expanded affine
30 transformation to inflate the intermediate results of the variable modulus operation to the

1 magnitude of the desired range, although only a portion of each resulting value is kept
2 because the unaltered series is linear and therefore predictable in nature.

3
4 Many of the existing pseudo-random number generators are computationally efficient and
5 generate well-distributed results. However, the recursive nature of the processes create
6 output results that exhibit strong linear correlation; this structure tends to make those
7 results exhibit characteristics which can be exploited to create predictions of future output
8 values. Predictability of the series may not be a problem for some applications, but still
9 indicates that the series is not as “random” for general applications as might be desired.

10
11 The invention described herein presents a general-purpose pseudo-random number
12 generator that offers output sequences with very long periods and very low predictability.

13 14 15 SUMMARY AND OBJECTS OF THE INVENTION

16
17 A primary object of the present invention is to provide a method and process for
18 generating pseudo-random numbers with very long period output sequences, well-
19 distributed actual output values, and very low predictability for general-purpose use.

20
21 Another object of the present invention is to introduce variable recursive matrix
22 operations into the pseudo-random number generator process where the transition
23 matrices are changed from one iteration of the generator to the next. The variations in the
24 transition matrices are determined by secondary pseudo-random number generators or
25 other processes where the secondary pseudo-random number generators or other
26 processes exhibit long cycles.

27
28 Another object of the present invention is to introduce variable recursive matrix
29 operations into the pseudo-random number generator process where the
30 offset augmentation matrices are changed from one iteration of the generator to the next.
31 The variations in the offset augmentation matrices are determined by secondary pseudo-

1 random number generators or other processes where the secondary pseudo-random
2 number generators or other processes exhibit long cycles.

3

4 Another object of the present invention is to introduce variable recursive matrix
5 operations into the pseudo-random number generator process where the modulus
6 operators are changed from one iteration of the generator to the next. The variations in
7 the modulus operators are determined by secondary pseudo-random number generators or
8 other processes where the secondary pseudo-random number generators or other
9 processes exhibit long cycles.

10

11 Another object of the present invention is to introduce a process for the use of multiple
12 modulus operators where the results are equally distributed across the range of actual
13 output values associated with the final modulus operator.

14

15 Another object of the present invention is to introduce the development of processes such
16 that the output matrix can be created in such a way as to be non-invertible, that is, having
17 no calculable inverse. The variable recursive matrix operations can be used to create
18 output sets that cannot be inverted making it impossible to determine constituent
19 components of the matrix operations simply from analysis of the observable output
20 results.

21

22 These objects are achieved by introducing a new type of pseudo-random number
23 generators that significantly extend the current state of the art. The multiple-recursive
24 matrix method (MRMM) framework that encompassed essentially all prior linear
25 methods is extended by this invention through the introduction of variable parameters.
26 The class of pseudo-random generators of the invention can be denoted as multiple
27 variable recursive matrix (MVRM) generators. As described in the following sections,
28 the new class of MVRM pseudo-random number generators of this invention is well
29 suited to general-purpose applications.

30

1 The need exists for pseudo-random number generators that offer results with more
2 random-like characteristics. The precise definition of “more random-like” is difficult to
3 specify, at best. This is especially true for pseudo-random number generators that are
4 purely deterministic, that is, those which can replicate the same output results exactly
5 given the same state of characteristics and input values for the generating process.
6 Essentially all of the widely used linear methods described above are deterministic
7 processes. The level of predictability is a reasonable indicator of the “randomness” of the
8 pseudo-random number generator. While the methods and processes of the invention
9 claimed herein are deterministic, the results are generally less predictable and more
10 “random” than those of other types of pseudo-random number generators.

11
12 Computational efficiency has often been a key determinant in the design of pseudo-
13 random number generators. However, computational power has increased dramatically
14 over the past years, making possible the introduction of pseudo-random number
15 generators that exchange reduced computational efficiency for increased “randomness”.
16 The pseudo-random number generators of the claimed invention offer just such a
17 compromise. Even so, depending on the specific implementation of the processes of the
18 invention, the decrease in computational efficiency may be relatively slight while the
19 gain in “randomness” may be substantial.

20
21 Pseudo-random number generators of the multiple-recursive matrix method (MRMM)
22 take the general form of $X_n = (A_1 X_{n-1} + \dots + A_k X_{n-k}) \bmod m$, where $A_1 \dots A_k$ and m are
23 constants and $X_{n-1} \dots X_{n-k}$ are the previous results of the process. The multiple variable
24 recursive matrix (MVRM) pseudo-random number generators of the claimed invention
25 take the general form of $X_n = ((A_{1,n} X_{n-1} + \dots + A_{k,n} X_{n-k} + B_{1,n} + \dots + B_{j,n}) \bmod m_{1,n}) \dots$
26 $\bmod m_{i,n}$, where:

27 | $A_{1,n} \dots A_{k,n}$, $B_{1,n} \dots B_{j,n}$, and $m_{1,n} \dots m_{i,n}$ are variable transition, offset augmentation
28 and modulus parameters for the n^{th} candidate output element of the matrix X ,
29 the transition matrices $A_{1,n} \dots A_{k,n}$ are created by secondary pseudo-random
30 number generators or other processes,

the offset augmentation matrices $B_{1,n} \dots B_{j,n}$ are created by secondary pseudo-random number generators or other processes, and

the modulus operators $m_{1,n} \dots m_{i,n}$ are created by secondary pseudo-random number generators or other processes.

The form of the processes is unchanged if the transition matrices $A_{1,n} \dots A_{k,n}$ are postmultiplied in the equation above instead of premultiplied as in the form shown.

In the MVRM process of the invention, the matrix of candidate output values X_n can be a matrix of any number of dimensions and sizes including columnar or row vector form.

The matrix will have a number of elements determined by the number of rows times the number of columns. The specific entries from the total elements contained in the matrix X_n to be used as the pseudo-random number generator candidate output values could be single elements from specific locations of the matrix or all the values of the entire matrix.

The dimensions of the candidate output matrix will determine the dimensions of the transition matrices and of the offset augmentation matrices. The transition matrices will be square matrices with row and column dimensions equal to the number of rows in the candidate output matrix. The offset augmentation matrices will have the same dimensions as the candidate output matrix.

The candidate output matrix X_n also can be created in such a way as to be non-invertible, that is, having no calculable inverse. This is a significant and distinguishing difference from the classic LCG pseudo-random number generators because the additive and multiplicative components of the LCG methods are always invertible, meaning that the LCG's observed output results are always invertible. Matrix and other similar data arrangements can be used to create output sets that cannot be inverted, making it impossible to determine constituent components of the matrix operations simply from analysis of the observed output results.

The multiple variable recursive matrix (MVRM) pseudo-random number generator process of the claimed invention has the form of $X_n = ((A_{1,n}X_{n-1} + \dots + A_{k,n}X_{n-k} + B_{1,n} + \dots$

$+ B_{j,n}) \bmod m_{1,n}) \dots \bmod m_{i,n}$, that is, the n^{th} value of the candidate output matrix is created by summing the multiple of the $n-1^{\text{th}}$ value of the candidate output matrix by the n^{th} value of the transition matrix A_1 (either premultiplied or postmultiplied) with all subsequent multiples through the multiple of the $n-k^{\text{th}}$ value of the candidate output matrix by the n^{th} value of the transition matrix A_k and the 1^{st} through the j^{th} values of the offset augmentation matrices B_n . The 1^{st} through the i^{th} modulus operators are sequentially applied to the resulting summation to yield the final n^{th} value of the candidate output matrix. The actual output pseudo-random numbers for that iteration of the generator are then taken from the candidate output matrix. In order to assure the uniformity of the distribution of the actual output values, certain results of the modulus operations may not be available for use as the pseudo-random number generator result; those intermediate results may or may not still be held in the historical sequence of candidate output matrix values X_n for the calculation of subsequent candidate output matrix values.

The variable transition matrices $A_{1,n} \dots A_{k,n}$ are determined by secondary pseudo-random number generators or other processes. For instance, a simple list of 100 possible values for A_1 could be compiled and the variation in the sequence of $A_{1,n}$ as n goes from 1 to 100 would consist of selecting the next entry from the list. As the list is exhausted, the selection would return to the beginning of the list. Similar lists could be used for A_2 through A_k with each variation being chosen from the sequences in the lists. Advantageously, the number of items in the lists could be chosen to be relatively prime, that is, no count of items in any of the lists would share a common factor with another. The length of the composite sequence created by this combined sequence of lists would have a cycle length equal to the product of the number of items in each list. Thus, with k set equal to 4 and list lengths of 100, 101, 103 and 107, the length of the cycle of combinations would equal 111,312,100 before the pattern of combinations would begin to repeat. Instead of lists, each $A_{1,n} \dots A_{k,n}$ could be determined by a secondary pseudo-random number generator with the cycle length of each pseudo-random number generator distinct from the others. The secondary pseudo-random number generators could be of virtually any form including the classical LCG or any of the variations mentioned above.

1 With distinct, relatively prime cycle lengths, the length of the composite sequence created
2 by the combined sequence of secondary pseudo-random number generators would have a
3 cycle length equal to the product of the separate cycle lengths. For example, with k set
4 equal to 4 and secondary pseudo-random number generator cycle lengths of 715,999,981,
5 714,673,789, 700,943,927 and 687,956,333, the length of the cycle of combinations
6 would equal 2.47×10^{35} before the pattern of combinations would begin to repeat.

7
8 The variable ~~offset~~augmentation matrices $B_{1,n} \dots B_{j,n}$ are determined by secondary
9 pseudo-random number generators or other processes. For instance, a simple list of 113
10 possible values for B_1 could be compiled and the variation in the sequence of $B_{1,n}$ as n
11 goes from 1 to 113 would consist of selecting the next entry from the list. As the list is
12 exhausted, the selection would return to the beginning of the list. Similar lists could be
13 used for B_2 through B_j with each variation being chosen from the sequences in the lists.
14 The number of items in the lists are ideally chosen to be relatively prime, that is, no count
15 of items in any of the lists would share a common factor with another. The length of the
16 composite sequence created by this combined sequence of lists would have a cycle length
17 equal to the product of the number of items in each list. Thus, with j set equal to 4 and
18 list lengths of 113, 109, 99 and 97, the length of the cycle of combinations would equal
19 118,280,151 before the pattern of combinations would begin to repeat. Instead of lists,
20 each $B_{1,n} \dots B_{j,n}$ could be determined by a secondary pseudo-random number generator,
21 ideally with the cycle length of each pseudo-random number generator distinct from the
22 others including those of the transition matrices A . The secondary pseudo-random
23 number generators could be of virtually any form including the classical LCG or any of
24 the variations mentioned above. With distinct, relatively prime cycle lengths, the length
25 of the composite sequence created by the combined sequence of secondary pseudo-
26 random number generators would have a cycle length equal to the product of the separate
27 cycle lengths. For example, with j set equal to 4 and secondary pseudo-random number
28 generator cycle lengths of 42,517,061, 43,477,631, 37,533,169 and 34,824,227, the
29 length of the cycle of combinations would equal 2.42×10^{30} before the pattern of
30 combinations would begin to repeat.

1 Using secondary pseudo-random number generators for the transition matrices A and also
2 for the offset augmentation matrices B with composite cycle lengths of 2.47×10^{35} and
3 2.42×10^{30} would yield a primary MVRM process pseudo-random number generator
4 with a cycle length of 5.96×10^{65} .

5
6 The modulus operators $m_{1,n} \dots m_{i,n}$ are determined by secondary pseudo-random number
7 generators or other processes. For instance, a simple list of 71 possible values for m_1
8 could be compiled and the variation in the sequence of $m_{1,n}$ as n goes from 1 to 71 would
9 consist of selecting the next entry from the list. As the list is exhausted, the selection
10 would return to the beginning of the list. Similar lists could be used for m_2 through m_i
11 with each variation being chosen from the sequences in the lists. The number of items in
12 the lists are ideally chosen to be relatively prime, that is, no count of items in any of the
13 lists would share a common factor with another. The length of the composite sequence
14 created by this combined sequence of lists would have a cycle length equal to the product
15 of the number of items in each list. Thus, with i set equal to 3 and list lengths of 71, 67
16 and 64, the length of the cycle of combinations would equal 304,448 before the pattern of
17 combinations would begin to repeat. Instead of lists, each $m_{1,n} \dots m_{i,n}$ could be
18 determined by a secondary pseudo-random number generator, advantageously with the
19 cycle length of each pseudo-random number generator distinct from the others including
20 those of the transition matrices A and of the offset augmentation matrices B. The
21 secondary pseudo-random number generators could be of virtually any form including
22 the classical LCG or any of the variations mentioned above. With distinct, relatively
23 prime cycle lengths, the length of the composite sequence created by the combined
24 sequence of secondary pseudo-random number generators would have a cycle length
25 equal to the product of the separate cycle lengths. For example, with i set equal to 3 and
26 secondary pseudo-random number generator cycle lengths of 7,337, 6,479 and 9,503, the
27 length of the cycle of combinations would equal 4.52×10^{11} before the pattern of
28 combinations would begin to repeat.

29
30 Use of secondary pseudo-random number generators for the modulus operators with
31 composite cycle lengths of 4.52×10^{11} in addition to secondary pseudo-random number

generators for the transition matrices A and for the ~~offset~~augmentation matrices B with composite cycle lengths of 2.47×10^{35} and 2.42×10^{30} , respectively, could yield a primary MVRM process pseudo-random number generator with a cycle length of 2.69×10^{77} . The cycle length of pseudo-random number generators with integrated varying modulus operators is difficult to evaluate since no theoretical basis for making such evaluation has yet been developed. However, because the system is composed of several independent elements each of which has quite long cycle lengths, the composite result could well be equivalent to the product of those cycle lengths leaving a very long resulting cycle length.

In order to assure the very long cycle lengths, an alternative form of the multiple variable recursive matrix (MVRM) pseudo-random number generator process of the invention could be used that has the form of $X_n = ((A_{1,n}X_{n-1} + \dots + A_{k,n}X_{n-k} + B_{1,n} + \dots + B_{j,n}) \bmod m_{1,n}) \dots \bmod m_{i,n}$ for the primary candidate output cycle component and the actual output values are generated through the multiple modulus operation $Z_n = (X_n \bmod r_{1,n}) \dots \bmod r_{g,n}$, that is, the n^{th} actual output value is generated by applying multiple varying modulus operators to the n^{th} value of the candidate output matrix. The initial modulus operators $m_{1,n} \dots m_{i,n}$ for the candidate output matrix could be chosen to accommodate the word-length constraints of the computer system and should advantageously be large prime numbers.

For either embodiment of the MVRM generator, the MVRM multiple modulus version or the alternative form described in the preceding paragraph, the modulus operators should ideally be chosen to be relatively prime to each other. The final modulus operator determines the range of the actual output values, e.g., choosing 256 as the final operator value creates a range of actual output values from 0 to 255. Other modulus operator values, whether chosen from lists, by secondary pseudo-random number generators, or by some other method, should fall into descending value order between the first operator (which should be the largest) and the final operator (which should be the smallest). All of the modulus operators should ideally be relatively prime to each other. Thus, if 256

were chosen as the final modulus operator, all of the other operators should be relatively prime odd numbers (to be relatively prime to 256 which is an even number).

In order to generate an equally and uniformly distributed set of actual output values, certain results from each of the modulus operation steps would have to be discarded according to the relationship of the modulus operators. For instance, if the value of $m_{1,n}$ mod $m_{2,n}$ was equal to 117, then 117 of the possible intermediate results would need to be discarded to assure the uniformity of the generated candidate or actual output distribution. Either the first 117, the final 117, or some arbitrary range of 117 of the possible intermediate output results could be discarded. To discard the first 117, the exclusion condition would be $X_n < 117$; to discard the final 117, the exclusion condition would be $X_n \geq m_{1,n} - 117$. The discarding process would be similarly applied to each subsequent set of modulus operations, e.g., $m_{2,n}$ mod $m_{3,n}$, $m_{3,n}$ mod $m_{4,n}$... $m_{i-1,n}$ mod $m_{i,n}$, where $m_{i,n}$ is the final modulus operator.

Another variation of the multiple modulus process would be to calculate $X_n = ((A^x_{1,n}X_{n-1} + \dots + A^x_{k,n}X_{n-k} + B^x_{1,n} + \dots + B^x_{j,n}) \bmod m^x_{1,n}) \dots \bmod m^x_{i,n}$ for the first primary candidate output cycle component and $Y_n = ((A^y_{1,n}Y_{n-1} + \dots + A^y_{k,n}Y_{n-k} + B^y_{1,n} + \dots + B^y_{j,n}) \bmod m^y_{1,n}) \dots \bmod m^y_{i,n}$ for the second primary candidate output cycle component and the actual output values would be generated through multiple modulus operations applied to the sum of X_n and Y_n as $Z_n = ((X_n + Y_n) \bmod m^z_{1,n}) \dots \bmod m^z_{i,n}$.

The process of the MVRM pseudo-random number generator could also be specified to assure that each candidate output value matrix of the form of $X_n = ((A_{1,n}X_{n-1} + \dots + A_{k,n}X_{n-k} + B_{1,n} + \dots + B_{j,n}) \bmod m_{1,n}) \dots \bmod m_{i,n}$ was a non-invertible matrix. This characteristic could be introduced by appropriate modification of the final offset augmentation matrix component $B_{j,n}$ to assure that the candidate output value matrix X_n was non-invertible. Were each of the candidate output value matrices non-invertible, then the multiplicative components created by the transition matrices (e. g., $A_{1,n}X_{n-1}$) would also be non-invertible regardless of the invertibility of the transition

1 | matrices A. However, the transition and offset augmentation matrices may themselves be
2 | non-invertible as contributing components of the resulting candidate output value matrix.

3
4 | All of the elements or only part of the elements of the candidate output value matrix X_n
5 | could be used as the actual output values of the pseudo-random number generator. Any
6 | remaining elements that are not used as pseudo-random number generator actual output
7 | values could be stored in the storage register and still contribute to the determination of
8 | subsequent candidate output value matrix results.

9
10 | The MVRM pseudo-random number generator claimed herein incorporates several
11 | components, each of which has distinct effects on the overall cycle length of the pseudo-
12 | random number generator process. In general, the use of long-cycle secondary pseudo-
13 | random number generators to determine the values of the transition matrices,
14 | offset augmentation matrices, and modulus operators should contribute to MVRM
15 | pseudo-random number generator cycles that are exceedingly long. The cycle length of
16 | pseudo-random number generators of the MVRM type is difficult to evaluate since no
17 | theoretical basis for making such evaluation has yet been developed. However, because
18 | the system is composed of several independent elements each of which has quite long
19 | cycle lengths, the composite result should be equivalent to the product of those cycle
20 | lengths leaving a very long resulting combined cycle length.

21
22 | An advantage of the present invention is that it presents a new unified framework for
23 | incorporating a large number of options into the pseudo-random number generator
24 | process creating nearly innumerable sets of alternative pseudo-random number
25 | sequences.

26 27 28 | BRIEF DESCRIPTION OF THE DRAWINGS

29
30 | FIG. 1 is a block diagram depicting the functional components of a MVRM pseudo-
31 | random number generator, according to the invention claimed herein.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

FIG. 2 is a block diagram depicting a general implementation of functional components of the MVRM pseudo-random number generator, according to the invention claimed herein.

FIG. 3 is a block diagram depicting the functional components of a MVRM pseudo-random number generator with both primary and secondary variable modulus reductions, according to the invention claimed herein.

FIG. 4 is a block diagram depicting a general implementation of functional components of the MVRM pseudo-random number generator with both primary and secondary variable modulus reductions, according to the invention claimed herein.

FIG. 5 is a block diagram depicting an implementation of the uniform variable modular reduction functional component of the MVRM pseudo-random number generator converting the intermediate output matrix X_{temp} to a uniformly distributed primary candidate output value matrix X_n , according to the invention claimed herein.

FIG. 6 is a block diagram depicting an implementation of the uniform variable modular reduction functional component of the MVRM pseudo-random number generator converting the primary candidate output value matrix X_n to a uniformly distributed secondary candidate output value matrix Z_n , according to the invention claimed herein.

FIG. 7 is a block diagram depicting a dual-sequence implementation of the MVRM pseudo-random number generator of the claimed invention, with a single variable modular reduction component.

FIG. 8 is a block diagram depicting an implementation of the MVRM pseudo-random number generator of the claimed invention, including an invertibility evaluation module for the creation of non-invertible candidate output value matrices.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to FIG. 1, a block diagram of the pseudo-random number generator system of the claimed invention is shown which incorporates a transition and offset augmentation summation process 11, a storage register 12 for initial and previously generated values of the primary candidate output matrix sequence X_n 3, a variable modular reduction process 13, a list or other process 14 for creating a value for transition matrix $A_{1,n}$, a list or other process 15 for creating values for all other transition matrices through $A_{k,n}$, a list or other process 16 for creating a value for offset augmentation matrix $B_{1,n}$, a list or other process 17 for creating values for all other offset augmentation matrices through $B_{j,n}$, a list or other process 18 for creating a value for modulus operator $m_{1,n}$, and a list or other process 19 for creating values for all other modulus operators through $m_{i,n}$. The values of the transition matrices $A_{1,n}$ 24 through $A_{k,n}$ 25 and of the offset augmentation matrices $B_{1,n}$ 26 through $B_{j,n}$ 27 along with the previously created or initial values of the primary candidate output matrices X_n 3 from the storage register 12 are provided to the transition and offset augmentation summation process 11 where they are aggregated through matrix multiplication and addition operations to create an intermediate value of the primary candidate output matrix shown as X_{temp} 2. The intermediate value X_{temp} 2 is then sent to the variable modular reduction process 13 where the modulus operators $m_{1,n}$ 28 through $m_{i,n}$ 29 are applied and resulting values evaluated for retention or removal to generate the primary candidate output matrix sequence X_n 3. The actual output values of the pseudo-random number generator X_{out} 1 are composed of all or some of the elements of the primary candidate output matrix X_n 3. Any remaining elements from the primary candidate output matrix X_n 3 that are not used as pseudo-random number generator actual output values X_{out} 1 could be stored in the storage register 12 and still contribute to the determination of subsequent primary candidate output matrix results.

In FIG. 2, one embodiment of the general pseudo-random number generator system of the invention is shown. The system shown in FIG. 2 details the transition and offset augmentation summation process 21 of the invention with the particular form X_{temp}

1 = $A_{1,n}X_{n-1} + \dots + A_{k,n}X_{n-k} + B_{1,n} + \dots + B_{j,n}$ using the transition matrices $A_{1,n}$ 24 through
 2 $A_{k,n}$ 25, the offset augmentation matrices $B_{1,n}$ 26 through $B_{j,n}$ 27, and the previously
 3 created or initial values of the primary candidate output matrices X_{n-1} through X_{n-k} from
 4 the storage register 12. The intermediate value X_{temp} 2 is then sent to the variable
 5 modular reduction process 23 with the form $X_n = ((X_{temp}) \bmod m_{1,n}) \dots \bmod m_{i,n}$ where
 6 the modulus operators $m_{1,n}$ through $m_{i,n}$ are applied and resulting values evaluated for
 7 retention or removal to generate the primary candidate output matrix sequence X_n 3. The
 8 retention/removal component of the variable modular reduction process when used to
 9 create uniformly distributed values is shown in more detail in FIG. 5.

10
 11 The actual output values X_{out} 1 of the pseudo-random number generator are composed of
 12 all or some of the elements of the primary candidate output matrix X_n 3. Any remaining
 13 elements from the primary candidate output matrix X_n 3 that are not used as pseudo-
 14 random number generator actual output values X_{out} 1 could be stored in the storage
 15 register 12 and still contribute to the determination of subsequent primary candidate
 16 output matrix results. The values for the transition matrices $A_{1,n}$ 24 through $A_{k,n}$ 25 are
 17 created by secondary pseudo-random number generators, are taken from pre-determined
 18 lists, or are created by other processes before being sent to the transition and
 19 offset augmentation summation process 21. The values for the offset augmentation
 20 matrices $B_{1,n}$ 26 through $B_{j,n}$ 27 are created by secondary pseudo-random number
 21 generators, are taken from pre-determined lists, or are created by other processes before
 22 being sent to the transition and offset augmentation summation process 21. The values for
 23 the modulus operators $m_{1,n}$ 28 through $m_{i,n}$ 29 are created by secondary pseudo-random
 24 number generators, are taken from pre-determined lists, or are created by other processes
 25 before being sent to the variable modular reduction process 23.

26
 27 In FIG. 3, an alternative embodiment of an implementation of the general pseudo-random
 28 number generator system of the invention is shown. The system shown in FIG. 3
 29 includes both primary variable modular reduction 31 and secondary variable modular
 30 reduction 32 components. As in the implementation shown in FIG.1, the process
 31 incorporates a transition and offset augmentation summation process 11; a storage register

12 for initial and previously generated values of the primary candidate output matrix sequence X_n 3; a primary variable modular reduction process 31; lists or other processes for creating transition matrices $A_{1,n}$ 14 through $A_{k,n}$ 15; lists or other processes for creating offset augmentation matrices $B_{1,n}$ 16 through $B_{j,n}$ 17; and lists or other processes for creating modulus operators $m_{1,n}$ 18 through $m_{i,n}$ 19. The transition and offset augmentation summation process 11 creates an intermediate value of the primary candidate output matrix X_{temp} 2. The intermediate value X_{temp} 2 is then sent to the primary variable modular reduction process 31 where the modulus operators $m_{1,n}$ 18 through $m_{i,n}$ 19 are applied and resulting values evaluated for retention or removal to generate the primary candidate output matrix sequence X_n 3. The primary candidate output matrix sequence X_n 3 is then sent to the secondary variable modular reduction process 32 where the modulus operators $r_{1,n}$ 38 through $r_{g,n}$ 39 are applied and resulting values evaluated for retention or removal to generate the secondary candidate output matrix sequence Z_n 33. The actual output values of the pseudo-random number generator X_{out} 1 are composed of all or some of the elements of the secondary candidate output matrix Z_n 33. The primary variable modular reduction process 31 may be implemented as a uniform variable modular reduction functional component as shown in FIG. 5 converting the intermediate output matrix X_{temp} 2 to a uniformly distributed primary candidate output value matrix X_n 3. Similarly, the secondary variable modular reduction process 32 may be implemented as a uniform variable modular reduction functional component as shown in FIG. 6 converting the primary candidate output value matrix X_n 3 to a uniformly distributed secondary candidate output value matrix Z_n 33.

In FIG. 4, one embodiment of the alternative implementation of FIG 3. is shown. As in the embodiment of FIG 2., the transition and offset augmentation summation process 21 of the invention takes the form $X_{temp} = A_{1,n}X_{n-1} + \dots + A_{k,n}X_{n-k} + B_{1,n} + \dots + B_{j,n}$ using the transition matrices $A_{1,n}$ 24 through $A_{k,n}$ 25, the offset augmentation matrices $B_{1,n}$ 26 through $B_{j,n}$ 27, and the previously created or initial values of the primary candidate output matrices X_{n-1} through X_{n-k} from the storage register 12. The intermediate value X_{temp} 2 is then sent to the primary variable modular reduction component 41 with the form $X_n = ((X_{temp}) \bmod m_{1,n}) \dots \bmod m_{i,n}$ to generate the candidate output matrix X_n 3.

Resulting values of the candidate output matrix X_n 3 are evaluated for retention or removal prior to storage in the storage register 12 to generate subsequent iterations of the primary candidate output matrix 3. The primary candidate output matrix X_n 3 also is sent to the secondary variable modular reduction process 42 with the form $Z_n = ((X_n) \bmod r_{1,n}) \dots \bmod r_{g,n}$ where the modulus operators $r_{1,n}$ 48 through $r_{g,n}$ 49 are applied and resulting values evaluated for retention or removal to generate the secondary candidate output matrix Z_n 33. The actual output values of the pseudo-random number generator X_{out} 1 are composed of all or some of the elements of the secondary candidate output matrix Z_n 33. Any remaining elements from the secondary candidate output matrix Z_n 33 that are not used as pseudo-random number generator actual output values X_{out} 1 are discarded. As in the embodiment of FIG 2., the values for the transition matrices $A_{1,n}$ 24 through $A_{k,n}$ 25 are created by secondary pseudo-random number generators, are taken from pre-determined lists, or are created by other processes before being sent to the transition and offset augmentation summation process 21. The values for the offset augmentation matrices $B_{1,n}$ 26 through $B_{j,n}$ 27 are created by secondary pseudo-random number generators, are taken from pre-determined lists, or are created by other processes before being sent to the transition and offset augmentation summation process 21. The values for the modulus operators $m_{1,n}$ 28 through $m_{i,n}$ 29 are created by secondary pseudo-random number generators, are taken from pre-determined lists, or are created by other processes before being sent to the primary variable modular reduction process 41. The values for the modulus operators $r_{1,n}$ 48 through $r_{g,n}$ 49 are created by secondary pseudo-random number generators, are taken from pre-determined lists, or are created by other processes before being sent to the secondary variable modular reduction process 42.

In FIG. 5, the retention and discarding procedures of the primary uniform variable modular reduction process are shown in detail. The intermediate value X_{temp} 2 is provided to the primary uniform variable modular reduction process 55. Each successive pair of modulus operators starting with $m_{1,n}$ 56 and $m_{2,n}$ 57 are used in the uniform variable modular processor 52 in the form $X_{temp2} = ((X_{temp}) \bmod m_{1,n}) \bmod m_{2,n}$. The uniformity of the distribution of the possible values of X_{temp2} over the range of 0 to $(m_{2,n}-1)$ is assured by discarding a certain number of candidate output values 53 from the

process. The number of values to be discarded is determined as $m_{1,n} \bmod m_{2,n}$ which would be a number greater than 0 if the modulus operators $m_{1,n}$ 56 and $m_{2,n}$ 57 were chosen to be relatively prime. The number of values to be discarded can be realized by discarding the first $m_{1,n} \bmod m_{2,n}$ elements of X_{temp} or by discarding the last $m_{1,n} \bmod m_{2,n}$ elements of X_{temp} . The process is successively repeated by providing each intermediate value to the primary uniform variable modular processor 52 for each successive pair of modulus operators. For example, the next successive pair of modulus operators ($m_{2,n}$ and $m_{3,n}$) would be used in the uniform variable modular processor 52 in the form $X_{temp3} = ((X_{temp2}) \bmod m_{2,n}) \bmod m_{3,n}$. However, since X_{temp2} was already created with the operation of $\bmod m_{2,n}$, the repetition of that step is unnecessary and simplifies to $X_{temp3} = (X_{temp2}) \bmod m_{3,n}$. As before, the uniformity of the distribution of the possible values of X_{temp3} over the range of 0 to $(m_{3,n} - 1)$ is assured by discarding the number of values determined as $m_{2,n} \bmod m_{3,n}$. The process is successively repeated by providing each intermediate value to the uniform variable modular processor 52 for each successive pair of modulus operators until the final set of $m_{i-1,n}$ 58 and $m_{i,n}$ 59 are used. In the final step, the uniform variable modular processor 52 has the form $X_{tempi} = ((X_{tempi-1}) \bmod m_{i-1,n}) \bmod m_{i,n}$ which again simplifies to $X_{tempi} = (X_{tempi-1}) \bmod m_{i,n}$. The uniformity of the distribution of the possible values of X_{tempi} over the range of 0 to $(m_{i,n} - 1)$ is assured by discarding a certain number of primary candidate output values 53 from the process. The number of values to be discarded is determined as $m_{i-1,n} \bmod m_{i,n}$ which is greater than 0 since $m_{i-1,n}$ 58 and $m_{i,n}$ 59 are relatively prime. The appropriate number of values to be discarded can be realized by discarding the first $m_{i-1,n} \bmod m_{i,n}$ elements of $X_{tempi-1}$ or by discarding the last $m_{i-1,n} \bmod m_{i,n}$ elements of $X_{tempi-1}$. The values of X_{tempi} in the final step are sent to the primary candidate output matrix X_n 50 as the results of the primary uniform variable modular reduction process 55.

In FIG. 6, the retention and discarding procedures of the secondary uniform variable modular reduction process are shown in detail. The primary candidate output matrix X_n 3 is provided to the secondary uniform variable modular reduction process 65. Each successive pair of modulus operators starting with $r_{1,n}$ 66 and $r_{2,n}$ 67 are used in the uniform variable modular processor 62 in the form $X_{secondary2} = ((X_n) \bmod r_{1,n}) \bmod r_{2,n}$.

1 The uniformity of the distribution of the possible values of $X_{\text{secondary}2}$ over the range of 0
 2 to $(r_{2,n} - 1)$ is assured by discarding a certain number of candidate output values 63 from
 3 the process. The number of values to be discarded is determined as $r_{1,n} \bmod r_{2,n}$ which
 4 should be a number greater than 0 since $r_{1,n}$ 66 and $r_{2,n}$ 67 are relatively prime. The
 5 number of values to be discarded can be realized by discarding the first $r_{1,n} \bmod r_{2,n}$
 6 elements of $X_{\text{secondary}}$ or by discarding the last $r_{1,n} \bmod r_{2,n}$ elements of $X_{\text{secondary}}$. The
 7 process is successively repeated by providing each intermediate value to the secondary
 8 uniform variable modular processor 62 for each successive pair of modulus operators.
 9 For example, the next successive pair of modulus operators ($r_{2,n}$ and $r_{3,n}$) would be used in
 10 the uniform variable modular processor 62 in the form $X_{\text{secondary}3} = ((X_{\text{secondary}2}) \bmod r_{2,n})$
 11 $\bmod r_{3,n}$. However, since $X_{\text{secondary}2}$ was already created with the operation of $\bmod r_{2,n}$,
 12 the repetition of that step is unnecessary and simplifies to $X_{\text{secondary}3} = (X_{\text{secondary}2}) \bmod$
 13 $r_{3,n}$. As before, the uniformity of the distribution of the possible values of $X_{\text{secondary}3}$ over
 14 the range of 0 to $(r_{3,n} - 1)$ is assured by discarding the number of values determined as $r_{2,n}$
 15 $\bmod r_{3,n}$. The process is successively repeated by providing each intermediate value to
 16 the uniform variable modular processor 62 for each successive pair of modulus operators
 17 until the final set of $r_{g-1,n}$ 68 and $r_{g,n}$ 69 are used. In the final step, the uniform variable
 18 modular processor 62 has the form $X_{\text{secondary}g} = ((X_{\text{secondary}g-1}) \bmod r_{g-1,n}) \bmod r_{g,n}$ which
 19 again simplifies to $X_{\text{secondary}g} = (X_{\text{secondary}g-1}) \bmod r_{g,n}$. The uniformity of the distribution
 20 of the possible values of $X_{\text{secondary}g}$ over the range of 0 to $(r_{g,n} - 1)$ is assured by discarding
 21 a certain number of secondary candidate output values 63 from the process. The number
 22 of values to be discarded is determined as $r_{g-1,n} \bmod r_{g,n}$ which is greater than 0 since
 23 $r_{g-1,n}$ 68 and $r_{g,n}$ 69 are relatively prime. The appropriate number of values to be
 24 discarded can be realized by discarding the first $r_{g-1,n} \bmod r_{g,n}$ elements of $X_{\text{secondary}g-1}$ or
 25 by discarding the last $r_{g-1,n} \bmod r_{g,n}$ elements of $X_{\text{secondary}g-1}$. The values of $X_{\text{secondary}g}$ in
 26 the final step are sent to the secondary candidate output matrix Z_n 60 as the results of the
 27 secondary uniform variable modular reduction process 65. The actual output values of
 28 the pseudo-random number generator $X_{\text{out}1}$ are composed of all or some of the elements
 29 of the secondary candidate output matrix Z_n 60.
 30

In FIG. 7, another alternative implementation of the general pseudo-random number generator system of the invention that includes two (or more) independent MVRM modules 71, 72 and a separate uniform variable modular reduction component 76 is shown in detail. Each of the independent MVRM modules 71, 72 operates as in the general version with the transition and offset augmentation summation process 11, the previously created values from the storage register 12, and the variable modular reduction process 13 creating the candidate output values X_n 73 or Y_n 74. The variable modular reduction process 76 accepts the independent candidate output values X_n 73 and Y_n 74 along with the variable modulus operators $m_{1,n}^z$ 77 through $m_{j,n}^z$ 78 to create the candidate output matrix Z_n 70 of the alternative implementation of the pseudo-random number generator. The specific actual output values X_{out} 1 are composed of all or some of the elements of the variable modulus candidate output matrix Z_n 70. The values for the modulus operators $m_{1,n}^z$ 77 through $m_{j,n}^z$ 78 are created by secondary pseudo-random number generators, are taken from pre-determined lists, or are created by other processes before being sent to the variable modular reduction process 76.

FIG. 8 portrays a particular embodiment of the general pseudo-random number generator system of the invention that includes a component assuring that the candidate output matrix X_n 80 cannot be inverted. FIG. 8 shows essentially the same system that was shown in FIG. 2 including details of the transition and offset augmentation summation process 21 with the form $X_{temp} = A_{1,n}X_{n-1} + \dots + A_{k,n}X_{n-k} + B_{1,n} + \dots + B_{j,n}$ using the transition matrices $A_{1,n}$ 24 through $A_{k,n}$ 25, the offset augmentation matrices $B_{1,n}$ 26 through $B_{j,n}$ 82, and the previously created or initial values of the primary candidate output matrices X_{n-1} through X_{n-k} from the storage register 12. However, unlike the system previously shown in FIG. 2, the non-invertible version of FIG. 8 includes an invertibility evaluation module 81 that evaluates the final offset augmentation matrix $B_{j,n}$ 82 and makes adjustments based on the characteristics of X_{temp} 2 not including $B_{j,n}$ 82 to assure that the result of the transition and offset augmentation summation process 21 yields a matrix that cannot be inverted. That non-invertible intermediate value of X_{temp} 2 is then sent to the variable modular reduction process 23 with the form $X_n = ((X_{temp}) \bmod m_{1,n}) \dots \bmod m_{j,n}$ where the modulus operators $m_{1,n}$ 28 through $m_{j,n}$ 29 are applied to

1 generate the primary non-invertible candidate output matrix sequence X_n 80. For
2 uniform variable modular reduction the retention/removal component of the process was
3 shown in more detail in FIG. 5. The actual output values of the pseudo-random number
4 generator X_{out} 1 are composed of all or some of the elements of the primary non-
5 invertible candidate output matrix X_n 80. Any remaining elements from the primary non-
6 invertible candidate output matrix X_n 80 that are not used as pseudo-random number
7 generator actual output values X_{out} 1 could be stored in the storage register 12 and still
8 contribute to the determination of subsequent primary non-invertible candidate output
9 matrix results. The values for the transition matrices $A_{1,n}$ 24 through $A_{k,n}$ 25 are created
10 by secondary pseudo-random number generators, are taken from pre-determined lists, or
11 are created by other processes before being sent to the transition and offset augmentation
12 summation process 21. The values for the offset augmentation matrices $B_{1,n}$ 26 through
13 $B_{j,n}$ 82 are created by secondary pseudo-random number generators, are taken from pre-
14 determined lists, or are created by other processes before being sent to the transition and
15 offset augmentation summation process 21 except that the final offset augmentation value
16 of $B_{j,n}$ 82 is evaluated and adjusted by the invertibility evaluation module 81 to assure
17 that the intermediate output matrix X_{temp} 2 cannot be inverted. The values for the
18 modulus operators $m_{1,n}$ 28 through $m_{j,n}$ 29 are created by secondary pseudo-random
19 number generators, are taken from pre-determined lists, or are created by other processes
20 before being sent to the uniform variable modular reduction process 23.

21
22 Although the present invention has been described in terms of the presently preferred
23 embodiment, it is to be understood that such disclosure is purely illustrative and is not to
24 be interpreted as limiting. Consequently, without departing from the spirit and scope of
25 the invention, various alterations, modifications, and/or alternative applications of the
26 invention will, no doubt, be suggested to those skilled in the art after having read the
27 preceding disclosure. Accordingly, it is intended that the following claims be interpreted
28 as encompassing all alterations, modifications, or alternative applications as fall within
29 the true spirit and scope of the invention.